REMARKS/ARGUMENTS

This paper responds to the Office Action of October 25, 2004, and requests reconsideration of the application. As noted in the accompanying "Request for Withdrawal of Finality," final rejection is premature, and this paper is entitled to entry as of right.

Claims 1-60, 63-115, and 118-135 are now pending, a total of 131 claims. Claims 1-19, 21-33, 37-47, 49-59, 61-85, 87-126 and 128-133 are not allowed; however as noted below, many of them are not rejected, either. Of the non-allowed claims, claims 22, 51, 63, 87, 94, 96 and 104 are independent.

The shortened statutory period runs through January 25, 2005. Accordingly, this response is timely.

The amendments to claims 87 and 92 are intended neither to clarify nor to change the scope of these claims, but rather to express the claims in language more idiomatic to the art.

I. Paragraphs 5-8: Incorporation by Reference

The material incorporated by reference into this application fully complies with all rules, including MPEP § 608.01(p)(A).

First, MPEP § 608.01(p)(B) states that there is no limit on material that may be incorporated from priority applications. As noted in the first sentence of this application, the applications incorporated by reference are priority applications, and thus under MPEP § 608.01(p)(B), any limitations on incorporation by reference "do not apply." Paragraph 7 of the Office Action has no legal basis.

Second, the Office Action reflects an incomplete reading of MPEP § 608.01(p)(A). § 608.01(p)(A) only limits incorporation of "essential material," and identifies "essential material" as "that which is necessary to (1) describe the claimed invention, (2) provide an enabling disclosure of the claimed invention, or (3) provide the best mode (35 U.S.C. 112)." § 608.01(p)(A) clarifies that the following are necessary elements of any objection:

- identify particular claims to which the material is "essential"
- identify the legal basis under which the material is essential – enablement, written description, best mode, or other

- identify the material that is "essential" to the identified claims

The Office Action never makes any averment that the material is "essential material" – if the material is not "essential," § 608.01(p)(A) states that such "nonessential subject matter may be incorporated by reference..." Further, without the three required showings, it is not at all clear that any requirement has any valid legal basis, and no applicant can make an informed decision on how to respond to a requirement. As a basic principle of administrative law, a requirement that does not permit a response has no legal existence.

Third, as the Office Action itself acknowledges, the material submitted on microfiche is not a "computer program listing." Therefore, new 37 C.F.R. § 1.96 is irrelevant, and does not control the form in which the material may be submitted. Further, the material submitted on microfiche exceeds the size of an appendix that may be submitted for printing under either old or new Rule 96, and it is not the type of information that may allowably be submitted on CD-ROM. In absence of any other appropriate medium, microfiche is an entirely proper way to place this material in the file for incorporation by reference unless and until it is shown to be "essential" to specific claims.

In absence of any breach of any rule, all incorporations by reference are proper.

II. Claim 22

Claim 22 is discussed at paragraph 52 of the Office Action of the February 2004 and paragraph 14 of the October 2004 Office Action. Claim 22 recites as follows:

> 22. A method, comprising the steps of:
>
> executing instructions fetched from first and second regions of a memory of a computer, the instructions of the first and second regions being coded for execution by computers following first and second data storage conventions, the memory regions having associated first and second indicator elements, the indicator elements each having a value indicating the data storage convention under which instructions from the associated region are to be executed;
>
> recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indictor element indicates the second data storage convention, and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

Paragraph 52.4 of the February Action compares the "the indicator elements each having a value indicating the data storage convention under which instructions from the associated region are to be executed" to Goetz '913, col. 17, lines 24-33 **and to no portion of any other reference**.

Applicant's paper of July 2004 fully responded to the February 2004 Action, by showing that this language is not met by the indicated portion of Goetz '913 (Response to Office Action of July 2004, at 34).

The October 2004 Office Action contains no showing that this claim language is met by Goetz '913 or by any other reference. The October Office Action apparently concedes that there is no correspondence between the indicated claim language and Goetz '913. The October Action does not indicate any other portion of any other reference that might be thought to correspond. The rejection has lapsed by operation of law, and Applicant is unable to respond further until the Examiner "Answers All Material Traversed" as required by MPEP § 707.07(f). No rejection has been raised in the October Action.

Without waiving the opportunity to reply to a properly-stated rejection if such rejection is presented in the future, and in an effort to advance prosecution, Applicant will make a best guess at the Examiner's view, and respond to that guess. Goetz '913, col. 17, lines 24-33 at best teaches only a P bit that indicates an instruction set. There is no indication that Goetz '913 ever uses two different "data storage conventions" as recited in claim 22, let alone indicates them with any "indicator." Rather, to the degree that Goetz '913 discusses "data storage convention" at all, he teaches that the data storage convention of one architecture is entirely abandoned: Goetz '913 teaches that the X86 page table format is "completely replaced" with his modified PowerPC page table format (col. 18, lines 63-64), so that the conventions of the two architectures match each other. This language of claim 22 renders claim 22 patentable over Goetz '913.

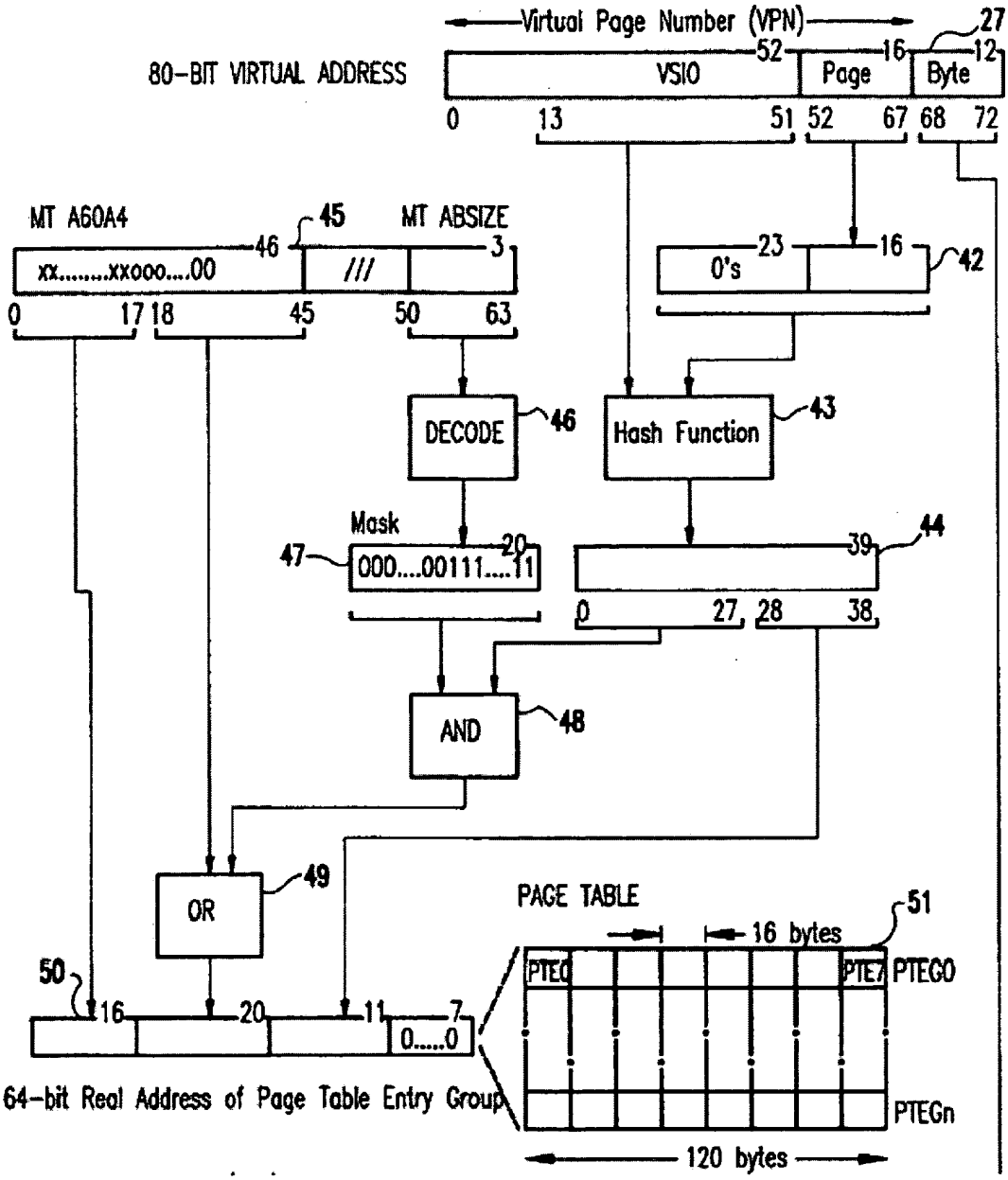
III. Claim 51

Claim 51 is discussed at paragraphs 29 and 30 of the February Office Action, and paragraph 14.2 of the October Office Action, in the context of Goetz '913 alone. Claim 51 recites as follows:

51. A method, comprising:

> storing instructions in pages of a computer memory managed by a virtual memory manager, the instruction data of the pages being coded for execution by, respectively, computers of two different architectures and/or under two different execution conventions;
>
> in association with pages of the memory, storing corresponding indicator elements indicating the architecture or convention in which the instructions of the pages are to be executed, the pages' indicator elements being stored in a table whose entries are indexed by physical page frame number;
>
> executing instructions from the pages in a common processor, the processor designed, responsive to the page indicator elements, to execute instructions in the architecture or under the convention indicated by the indicator element corresponding to the instruction's page.

Claim 51 recites that the pages' indicator elements are stored in "a table whose entries are indexed by physical page frame number. In contrast, the Office Action points to Goetz '913, Fig. 4, and col. 10, lines 4-28, which read (in pertinent part) as follows:



.... In the PowerPC architecture as shown in FIG. 4, VAs [virtual addresses] in register 27 are translated to physical addresses (PAs) via a hashed page table search. More particularly, bits 52 to 67 (page field) of the 80-bit virtual address in register 27 [and combined with other values] to form ... page table origin register 50. ... The highest seven bits of register 50 are forced to zero to form the real address of the page table 51 in memory. The individual page table group entries are searched until an entry 52 is found whose virtual segment ID matches that of the original VA. When found, the real page number is extracted from the page table group entry 52...

This portion of Goetz '913 clearly states four things that are incompatible with the interpretation expressed in the Office Action:

(a) Register 50 contains the real address of only the "page table" 51 as a whole, and is never used to "index" an "entry," as recited in claim 51.

(b) To find a particular entry in the page table (as recited in claim 51) Goetz '913 uses various portions of the "virtual" address, not any value related to any "physical" address, as recited in claim 51.

(c) Goetz relies on the difference between "real" addresses (or "physical address," as recited in the claim) and "virtual" addresses for correct behavior.

(d) Goetz '913 "extracts" the "real page number" from the page table entry that was located based on its "virtual segment ID." The contrary view expressed in ¶ 14.2 of the October Office Action, that the "real address" of a page is used to "access a particular element" from which Goetz then "extracts" the real page number of the very page whose entry is indexed, is circular. Paragraph 14.2 makes no engineering sense. The scheme proposed in the October Office Action is essentially the same as looking in the white pages of the phone book to find the correct spelling of a person's last name, using the known correct spelling of the last name to do the lookup. No person would rationally do such a thing, or design such a step into a computer – clearly Goetz '913 does not teach the technique set out in the Office Action.

The October Office Action further confuses the issues in several other respects. Is the Examiner's position that the conventional distinction between "physical" and "virtual" addresses may be ignored? Is the Examiner's position that the conventional definition of "indexing of entries" of a table may be ignored? The confusion is exacerbated by the Office Action's partial and misleading quotation from the Microsoft Computer Dictionary. In pertinent part, the relevant Microsoft definition of "index" reads as follows:

> **index** ... In programming, a scalar value that allows direct access into a multielement data structure such as an array. The index allows the programmer to calculate or otherwise derive the location of the desired element, eliminating the need for a "brute-force" [to separate the collection of] elements. ...

The "elements" of the second sentence of this definition are directly relevant to the "entries" recited in the claim. Because this second sentence was entirely omitted from the Office Action, the Examiner's view with respect to the "elements" of the definition, or the "entries" of claim 51, cannot be discerned. Until a clear, internally-consistent position is stated, Applicant is unable to meaningfully respond.

Claim 51 recites a limitation absent from Goetz '913, and is therefore patentable over Goetz '913.

IV. Claim 63

Claim 63 is discussed at paragraphs 32 and 34 of the February 2004 Office Action, and paragraph 14.3 of the October 2004 Office Action, in the context of Goetz '913 alone. Claim 63 recites as follows:

> 63. A microprocessor chip, comprising:
>
> an instruction unit, configured to fetch instructions from a memory managed by the virtual memory manager, and configured to execute instructions coded for first and second different computer architectures or coded to implement first and second different data storage conventions;
>
> the microprocessor chip being designed (a) to retrieve indicator elements stored in association with respective pages of the memory, each indicator element indicating the architecture or convention in which the instructions of the page are to be executed, and (b) to recognize when instruction execution has flowed or transferred from a page of the first architecture or convention to a page of the second, as indicted by the respective associated indicator elements, and (c) to alter a processing mode of the instruction unit or a storage content of the memory to effect execution of instructions in accord with the indicator element associated with the page of the second architecture or convention;
>
> wherein the indicator elements are stored in a table distinct from a primary address translation table and from portions of the primary address translation table cached in a TLB (translation lookaside buffer) used by a virtual memory manager, the indicator elements of the table being stored in association with respective pages of the memory.

The October Office Action points to Goetz' page table 51, 52 and TLB as potentially corresponding to the "table" with "indicator elements" of the last paragraph of claim 63. However, claim 63 expressly disclaims these two locations. Claim 63 is therefore patentable over Goetz '913.

V. Claim 87

Claim 87 is briefly mentioned in paragraph 100 of the Office Action of February 2004, in relation to some unspecified portions of Goetz '913, Brender '422, and Murphy '947, combined in some unspecified way. The February Action made no element-by-element comparison of claim 87 to the prior art, and was no more than an invitation to Applicant to attempt to guess at

the ground of rejection. A single limitation of claim 87 is discussed in paragraph 14.4 of the October 2004 Action.[1] Because the claim as a whole has never been compared to the prior art element-by-element, no rejection exists.

Claim 87 recites as follows:

> 87. A method, comprising the steps of:
>
> executing a control-transfer instruction in the hardware of a computer under a first execution mode of the computer, the instruction being architecturally defined to transfer control directly to a destination instruction for execution in the hardware of the computer in a second execution mode of the computer;
>
> before executing the destination instruction, altering the data storage content of the computer to establish a program context under the second execution mode that is logically equivalent to the context of the computer as interpreted under the first execution mode, the reconfiguring including at least one data movement operation not included in the architectural definition of the control-transfer instruction.

Neither Office Action has ever addressed the language of claim 87, an "instruction" having certain properties. Instead, the Office Action notes that Brender '422 teaches transferring control to a "routine." The Office Action never mentions an "instruction" in the context of claim 87. The Office Action is irrelevant to claim 87.

Further, the October Office Action misstates the content of Brender '422. Brender '422 makes clear that no "instruction" meeting claim 87 can exist.

First, Brender '422, col. 8, lines 31-37 and col. 10, line 43 through col. 15, line 16 teaches that his compiler identifies every transfer of control between different instruction sets, and then inserts a software "jacket" at that transfer. That "jacket" is interposed between the source and destination "instruction" – there is never a "direct" transfer at an instruction level, as recited in claim 87.

Second, Brender '422 then absolutely ensures that there is no possibility of a direct transfer from one instruction set to the other, except through the jacket: before "Y" routines are linked with "X" routines ("X" and "Y" being the two instruction set architectures discussed in Brender '422), "The names of the called [Y] routines are hidden at link time..." (col. 15, lines 42-44). Because the names of the "Y" routines are hidden before they are linked with "X"

---

[1] The claim as a whole has never been compared to the art at issue; no rejection exists.

routines, the linker cannot possibly link an "X" instruction to "transfer control directly to a destination instruction" in the "Y" instruction set. Instead, Brender '422 ensures that the software jacket is executed on every transfer.

Third, Brender '422 teaches that any control transfer instructions that are not routed through a jacket are illegal. Brender '422 teaches that these instructions are architecturally defined to fault, not to "transfer control" as recited in claim 87. Paragraph 14.4 of the October Office Action quotes one of the clearest of these statements, Brender '422, col. 15, lines 48-50:

> In the present embodiment, it is based on the fact that a direct X call to a Y routine incurs an X operand fault.

This sentence from Brender '422 teaches exactly opposite the interpretation suggested in the Office Action: an attempted "direct call" is architecturally defined to "fault," that is, to not complete at all. Such an instruction is not architecturally defined to "transfer control directly" to a destination, as recited in claim 87.

Because the Office Action does not show any correct correspondence between this language of claim 87 and any prior art, claim 87 may be allowed.

VI. Claim 94

Claim 94 is discussed at paragraphs 43-44 of the Office Action of February 2004, and in paragraph 14.5 of the October 2004 Action, in the context of Brender '422 alone (with Murphy '947 incorporated by reference). The claim as a whole has never been compared element-by-element to the art at issue; no rejection exists.

Claim 94 recites as follows:

> 94. A method, comprising the steps of:
>
> executing a section of computer object code twice, without modification of the code section between the two executions, the code section materializing a destination address into a register and being architecturally defined to directly transfer control indirectly through the register to the destination address, the two executions materializing two different destination addresses;
>
> the two destination code sections at the two materialized destination addresses being coded in two distinct instruction sets and, respectively, obeying the default calling conventions native to each of the two instruction sets, neither instruction set being a subset of the others.

Even taken together, the discussion in the two Office Actions is too incomplete to constitute a rejection. For example, the claim recites a "register" that performs certain functions. Neither Office Action indicates any "register" that is thought to perform these functions. No rejection exists.

Brender '422 teaches away from the "directly transfer control indirectly through the register to the destination address [of] code ... being coded in two distinct instruction sets and, respectively, obeying the default calling conventions native to each of the two instruction sets" of claim 94. The paragraph immediately between the two paragraphs cited in the Office Action, Brender '422, col. 21, lines 50-57, reads as follows:

> Indirect calls from Y code are made to work, even when the target routine might be in a X domain, using a special code generation convention in certain compilers. Indirect calls are **always** performed by invoking a run-time routine ...

Then, in the paragraph following the second paragraph cited in the Office Action, Brender '422 father clarifies:

> The only change is to replace the load of R26 from the second quadword of the target procedure descriptor with a load of the code address for [the runtime routine that implements cross-ISA calls]. The code address for the [runtime routine] is loaded as opposed to a procedure value address; [the runtime routine] is invoked with a non-standard call and there is no associated procedure descriptor.

Brender '422 makes clear that the "destination address" materialized into register R26 is "always" the address of the special jacketing run-time routine, not the "destination address [of] code ... being coded in two distinct instruction sets and, respectively, obeying the default calling conventions native to each of the two instruction sets" as recited in claim 94.

The above discussion of claim 87 further clarifies that Brender '422 expressly teaches away from a "direct" transfer of control between "two distinct instruction sets" as recited in claim 94.

Claim 94 is neither rejected nor rejectable over Brender '422.

VII. Claim 96

Claim 96 is briefly mentioned, without being rejected, in paragraph 100 of the Office Action of February 2004, over unspecified portions of Goetz '913, Brender '422, and Murphy

'947, combined in some unspecified way. A single limitation of claim 96 is discussed in paragraph 14.4 of the October 2004 Action. The claim as a whole has never been compared element-by-element to the art at issue; no rejection exists.

Claim 96 recites as follows:

> 96. A microprocessor chip, comprising:
>
> two instruction decoders designed to decode instructions of first and second instruction sets, respectively, and circuitry of a single instruction pipeline designed to execute the instructions decoded by either of the two instruction decoders;
>
> circuitry designed to detect when execution flows or transfers control from code coded in one instruction set to code coded in the other, program code in the first and second instruction sets using first and second different data storage conventions, respectively; and
>
> circuitry and/or software designed to respond to the detection by altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

Because neither Office Action designates "the particular part [of any reference] relied on ... as nearly as practicable" that might correspond to the underlined paragraph of claim 96, and neither "clearly explains" the pertinence of any reference to that paragraph (in violation of 37 C.F.R. § 1.104(c)(2)), Applicant can only guess at the Examiner's view, and attempt to respond to that guess. Without waiving the opportunity to reply to a proper rejection if such be presented in some future Office Action, and in an effort to advance prosecution, Applicant notes as follows:

- Neither Brender '422 nor Murphy '947 suggest "circuitry" related to the function recited in that paragraph – both are implemented entirely in software.
- As discussed in more depth in § II (discussion of claim 22), *supra*, Goetz '913 at best discusses circuitry for detecting a change of instruction set, where the two instruction sets use identical data storage conventions. Goetz '913 teaches away from the invention, by teaching that the normal "data storage convention" associated with one of the instruction sets must be abandoned in favor of the "data storage convention" of the other, in order to result in a workable system.

Because claim 96 recites "circuitry" absent from all prior art references, claim 96 may be allowed.

Paragraph 14.6 of the October Office Action attempts to show "motivation to combine" by showing that "software and hardware are logically equivalent and interchangeable." This paragragh of the Action misstates the law in two respects, and misstates the engineering realities.

First, MPEP § 2143.03 states that there is no obviousness unless every element of the claim is shown to exist in the prior art. *Motorola v. Interdigital Technology Corp.*, 121 F.3d 1461, 1466-67, 43 USPQ2d 1490, 1490-91 (Fed. Cir. 1997) (reversing a jury verdict of obviousness because an element was not taught in the particular art relied upon, even though that element was known elsewhere). "Logically equivalent" is nowhere authorized as an alternative test for obviousness.

Second, paragraph 14.6 is no more than a statement that "references can be combined or modified," reasoning that is forbidden at MPEP § 2143.01. Paragraph 14.6 exceeds the authority delegated to an examiner.

Third, Brender's and Murphy's software techniques cannot practically be implemented in hardware: both Brender '422 and Murphy '947 require frequent modification that is not practical in hardware. Brender '422 teaches that an entire program must be compiled and linked, if any single instruction changes, on the same basis as any other program. Brender '422, col. 7-16, see esp. col. 15, lines 24-57 (both "manual" and "automatic" jacketing require modification of the binary image at "image build time"); Murphy '947, col. 7, lines 6-15. Hardware cannot be generated on such a "throw-away" basis in any cost-effective manner – for example, any such attempt would require several weeks of turn-around time and (typically) hundreds of thousands of dollars. Instead, hardware is designed to be general, so that it need not be re-fabricated for each revision. Accordingly, Applicants' specification discloses a few simple, general structures and techniques that can be adapted to allow cross-instruction-set calls in many different programs. See, e.g., sections I-IV of the specification, pages 31-75.

A second key difference between hardware and software is cost. The cost of software goes up roughly linearly with capability. In contrast, the cost of hardware goes up roughly as the cube of die size, or roughly as the cube of chip capability. See lecture slides from the University of California at Berkeley, available at vco.ett.utu.fi/courses/ETT_2015/kalvot/luento4.pdf, and attached hereto as Exhibit A. Therefore, most CPU chip designs avoid large on-chip structures

that are not directly related to the core tasks of executing instructions. Because Brender '422 and Murphy '947 are entirely software based, they can use large structures that were not amenable to hardware implementation at the time of the invention. For a first example, Brender '422 and Murphy '947 teach that an entire program is linked together for execution. Brender '422, col. 15, lines 24-57; Murphy '947, col. 7, lines 6-40. Implementing an entire program in hardware would not be recognized as desirable by one of ordinary skill. For a second example, Murphy's and Brender's "jacketing tables" and "jacketing routines" may be quite large, because a specific data structure must be generated for each routine that could conceivably be called from the opposite instruction set. E.g., Murphy '947, col. 6, lines 4-8 ("an array that forms a signature ... for each incoming call in each Y code routine..."); Brender '422, col. 8, lines 31-37 ("Jacketing requires that knowledge ... for each subprogram in each domain...); Brender '422, col. 10, line 43 through col. 15, line 16. Because Brender '422 and Murphy '947 use entirely-software-based approaches, they have no motivation to teach techniques that could practically be implemented in hardware. Implementing their large, non-general structures in hardware would not be practical, let alone desirable, to one of ordinary skill.

Claim 96 recites "circuitry" absent from the art now of record. No rejection would be proper.

VIII. Claim 104

Claim 104 was briefly mentioned in paragraphs 10 and 42 of the Office Action of February 2004 in the context of **Goetz '913 alone**. Because the Office Action of February 2004 made no element-by-element comparison of claim 104 to Goetz '913 or any other reference, no rejection was raised in February 2004.

Paragraph 14.7 of the Office Action of October 2004 states that "one cannot show nonobviousness by attacking references individually." The October Action is a *non sequitur* – no obviousness was raised against claim 104 in February 2004, and thus an attack against the single reference applied is a proper and complete response.

Without waiving the opportunity to reply to a properly-stated rejection if such rejection is presented in the future, and in an effort to advance prosecution, Applicant makes a best guess at the Examiner's view, and observes as follows. Claim 104 recites:

> 104 A method, comprising the steps of:
>
> executing instructions fetched from first, second and third regions of a single address space of the memory of a computer, the instructions of the first and second regions being coded for execution by computers of first and second architectures, the instructions of the second and third regions following first and second data storage conventions, respectively, the memory regions having associated modifiable indicator elements, a hardware structure for storing the indicator elements enforcing a requirement that the memory regions be necessarily disjoint, the modifiable indicator elements each having a value indicating the architecture and data storage convention under which instructions from the associated region are to be executed;
>
> when execution of the instruction data flows or transfers between the first, second and third regions, adapting the computer for execution in the architecture and/or convention of the region transferred to.

Claim 104 recites two memory regions that "[follow] first and second data storage conventions." As discussed above in connection with claim 22, at page 41, there is no indication that Goetz '913 uses two different "data storage conventions." Indeed, it appears that Goetz deliberately avoids this. Accordingly, claim 104 is not anticipated by Goetz '913.[2]

IX. The IDS of July 15, 2004

Paragraph 4 of the October Office Action states that many references cited in the IDS of July 2004 were not considered because no legible copies were provided. The Office Action is simply wrong. Legible copies of each reference of which copies were required were filed in prior applications, as cataloged in the accompanying Request for Withdrawal of Finality.

A replacement IDS accompanies the copy of this paper submitted by First Class Mail.

---

[2] Applicant notes that claim 104 recites two memory regions that are "necessarily disjoint." Claims 104 and 113 are the only two claims that recite this limitation. Neither Office Action mentions this limitation, and neither states where "the same reasons set fourth [sic] in the previous claim rejections, supra" can be found. Thus, neither Office Action states any rejection of claim 104.

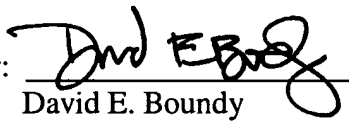
X. Conclusion

In view of the amendments and remarks, Applicant respectfully submits that the claims are in condition for allowance. Applicant requests that the application be passed to issue in due course. The Examiner is urged to telephone Applicant's undersigned counsel at the number noted below if it will advance the prosecution of this application, or with any suggestion to resolve any condition that would impede allowance. In the event that any extension of time is required, Applicant petitions for that extension of time required to make this response timely. Kindly charge any additional fee, or credit any surplus, to Deposit Account No. 23-2405, Order No. 114596-03-4000.

Respectfully submitted,

WILLKIE FARR & GALLAGHER LLP

Dated: January 25, 2004

By: 

David E. Boundy
Registration No. 36,461

WILLKIE FARR & GALLAGHER LLP
787 Seventh Ave.
New York, New York 10019
(212) 728-8000
(212) 728-8111 Fax

Exhibit A to

Response to Office Action

CS152
Computer Architecture and Engineering

Lecture 5: Cost and Design

September 10, 1997

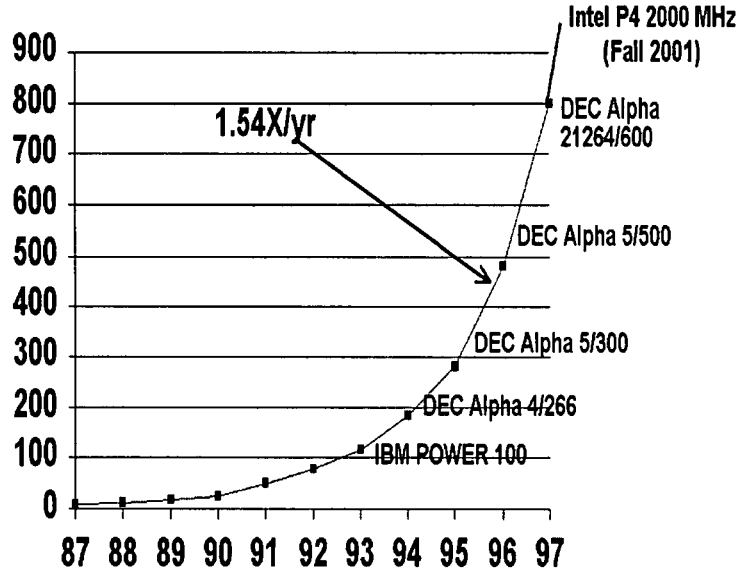
Dave Patterson (http.cs.berkeley.edu/~patterson)

lecture slides: http://www-inst.eecs.berkeley.edu/~cs152/

Review: Performance and Technology Trends

- Technology Power: 1.2 x 1.2 x 1.2 = 1.7 x / year
  - Feature Size: shrinks 10% / yr. => Switching speed improves 1.2 / yr.
  - Density: improves 1.2x / yr.
  - Die Area: 1.2x / yr.
- The lesson of RISC is to keep the ISA as simple as possible:
  - Shorter design cycle => fully exploit the advancing technology (~3yr)
  - Advanced branch prediction and pipeline techniques
- Bigger and more sophisticated on-chip caches

Technology Trends: Processor Performance

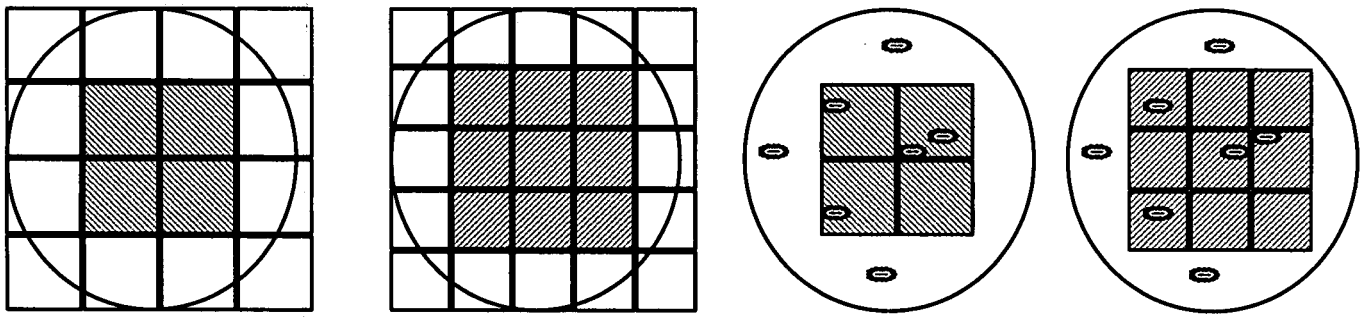


Processor performance increase/year, referred to as Moore's Law (transistors/chip)

Integrated Circuit Costs

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} * \text{Die yield}}$$

$$\text{Dies per wafer} \sim \text{eff} \frac{\text{Wafer Area}}{\text{Die Area}}$$



$$\text{Die Yield} = \frac{\text{Wafer yield}}{\left\{ 1 + \frac{\text{Defects\_per\_unit\_area} * \text{Die\_Area}}{?} \right\}^{?}}$$

Die Cost is goes roughly with the cube of the area.